An Optimization Method for Cascaded Filters

By SHLOMO HALFIN

(Manuscript received August 1, 1969)

This paper presents a procedure for decomposing an nth *order filter into cascadable second order sections. The procedure is optimal in that it minimizes the maximal response range for the sections within the frequency band of interest. The procedure, based on a modified version of the Bottleneck Assignment Algorithm, describes methods of listing all the optimal decompositions as well as of finding a special "nested" optimal decomposition.*

I. INTRODUCTION

Let $\phi(s)$ be a transfer function

$$\phi(s) = \frac{f(s)}{g(s)}$$

where f and g are polynomials with real coefficients, and the degree of $f \leqq$ degree of g.

We consider all the decompositions of the form $\phi(s)=\phi_1(s)\phi_2(s) \cdots \phi_t(s)$ where

$$\phi_i(s) = \frac{f_i(s)}{g_i(s)} \tag{1}$$

$f_i(s)$ and $g_i(s)$ are real polynomials and the degree of f_i does not exceed the degree of g_i. The g_i are quadratic polynomials, except when the degree of g is odd; then one g_i is linear.

Let L be a passband region for ϕ, where L is a finite union of passband intervals. Then for every ϕ_i, a number $d(\phi_i)$ is defined by

$$d(\phi_i) = 20 \log_{10} \left[\frac{\underset{\omega \in [0,\infty)}{\text{Max}} \mid \phi_i(j\omega) \mid}{\underset{\omega \in L}{\text{Min}} \mid \phi_i(j\omega) \mid} \right]. \tag{2}$$

Also let

$$d = \max_{i=1,\ldots,t} d(\phi_i). \tag{3}$$

Then d is a function of the decomposition. We present a procedure that determines the decomposition(s) with a minimal d. E. Lueder proposed this optimality criterion.[1]

II. METHOD

First, we artificially equate the number of zeros [zeros of $f(s)$], and poles [zeros of $g(s)$], by adding a suitable number of "zeros at infinity" corresponding to constant unit polynomials. Next we make this mutual number even by adding a zero and a pole at infinity, if necessary. In this way we get, say, $2t$ zeros and $2t$ poles.

Pairing two zeros creates an f_i ; a real zero can be paired with any other real zero, while a complex zero must be paired with its conjugate in order to get a real f_i . The same is true for creation of g_i by pairing of poles.

In the following we assume that all poles, except perhaps one, are complex and therefore fixed paired. We call the real zeros which are not fixed paired *free zeros*.

Next we make all possible pairings of the free zeros. Each such pairing, together with the fixed pairing, decomposes $f(s)$ and $g(s)$:

$$f(s) = f_1(s)f_2(s) \cdots f_t(s);$$
$$g(s) = g_1(s)g_2(s) \cdots g_t(s).$$

Then we compute the matrix $D = (d_{ik})$, where the elements

$$d_{ik} = d\left(\frac{f_i}{g_k}\right) \tag{4}$$

are computed from definition (2). The element d_{ik} represents the "cost" of *matching* zero-pair i with pole-pair k.

An *assignment* is a feasible set of matchings. Using the Bottleneck Assignment Algorithm, we determine an assignment $k_1 , \cdots , k_t$ for which

$$\max_{i=1,\ldots,t} d_{ik_i}$$

will be minimal. We call this minimum the *optimal d value* for this pairing of free zeros. Going through all the possible pairings of free zeros, we find an *optimal pairing* which yields the smallest optimal d value.

Since an optimal assignment (for a given optimal pairing) is usually

not unique, procedures for ohtaining all the optimal solutions (assignments) or a *nested* solution are given. A *nested* solution is ohtained hy taking an optimal solution, fixing the matching with the largest d value, and then proceeding to look for an optimal assignment for the remaining $t-1$ f_i's and $t-1$ g_k's, and so on.

III. THE BOTTLENECK ASSIGNMENT ALGORITHM

This section discusses the Bottleneck Assignment Algorithm and its adaptation to the present prohlem.

Let $U = (u_{i,j})$ he a real $t \times t$ matrix. A matching is an ordered pair of integers (i_k , j_k) $1 \leqq i_k \leqq t$, $1 \leqq j_k \leqq t$. We associate with the matching (i_k , j_k) the corresponding cell in U. The element in this cell $u_{i,j}$ is called the cost of the matching.

A set $A = \{(i_k , j_k); k = 1, \cdots , t\}$ of t matchings (cells) is called an *assignment* if in every row and in every column of U there is a cell that helongs to A. The hottleneck assignment prohlem looks for an assignment which minimizes the maximum of its matchings' costs.

The Gross algorithm[2], is hased on the following iterative step:

(*) An assignment A and a real numher α, which does not exceed all the costs of A, are given; then either a new assignment A' is con-

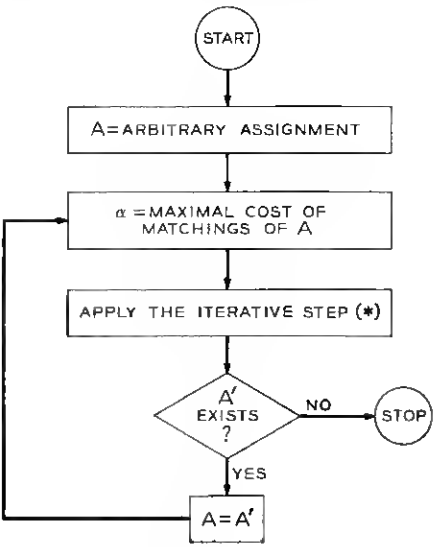

Fig. 1 — Flow chart for solving the bottleneck assignment problem.

structed, so that α exceeds more costs in A' than it does in A, or it is established that no such A' exists.

The flow chart in Fig. 1 solves the bottleneck assignment problem. This algorithm is fast and requires small memory space, since only the present assignment must be stored.

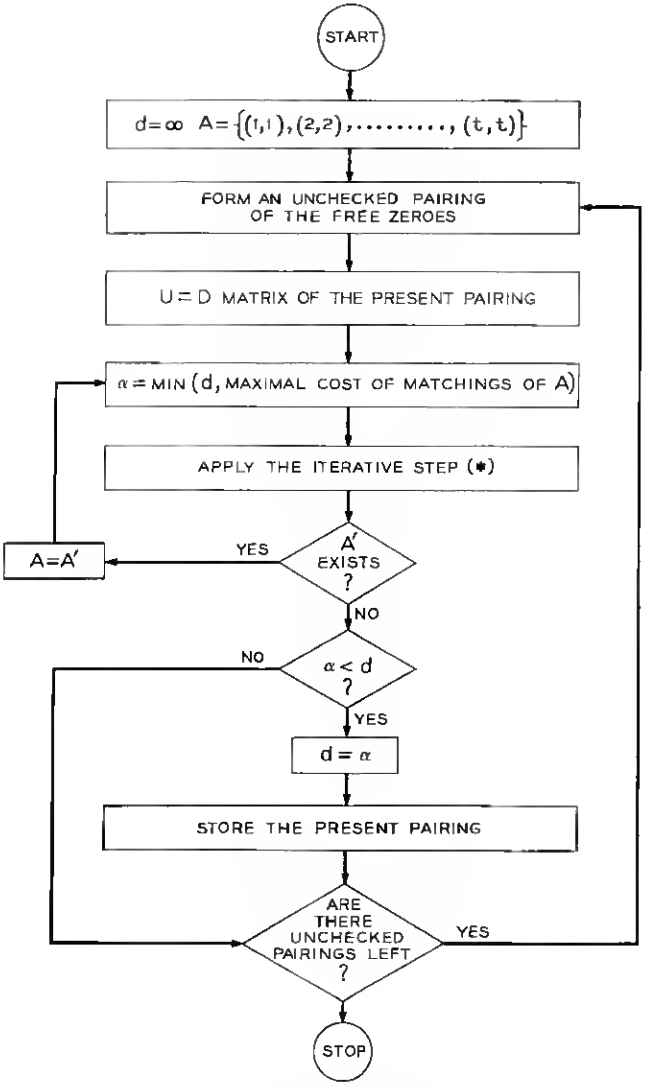

Fig. 2 — Flow chart for finding the optimal pairing and its optimal d-value.

The basic algorithm was modified to find the optimal pairing of the free zeros and its optimal d-value (Fig. 2). Note that the value of α in every iterative step (*) does not exceed the minimum of the optimal d values of the checked pairings. Thus any pairing that does not reduce the d value already obtained is immediately disregarded.

Also, for each pairing we use the optimal assignment of the preceding pairing, as an initial assignment. Thus the costs of the initial assignment matchings that correspond to the fixed paired zeros do not exceed the current d value. These procedures considerably reduce the amount of computation required for finding the optimal pairing and its optimal d value.

IV. CREATION OF THE NESTED SOLUTION

Let U denote the cost matrix which corresponds to the optimal pairing. The nested solution is created by successively applying the bottleneck assignment algorithm t times and modifying U each time in such a way that the matching with the largest cost becomes fixed and irrelevant in the further computations.

Let (i_k^*, j_k^*) be the matching with the largest cost at a certain stage. Then (i_k^*, j_k^*) becomes a part of the nested solution. U is then modified as follows:

$$U_{i_k^*,s} = \infty \quad \text{for all} \quad s \neq j_k^*;$$

$$U_{s,j_k^*} = \infty \quad \text{for all} \quad s \neq i_k^*;$$

$$U_{i_k^*,j_k^*} = 0.$$

It is easy to verify that this modification has the properties described.

V. A COMPUTATIONAL METHOD TO GENERATE ALL THE OPTIMAL ASSIGNMENTS

Let U denote again the cost matrix which corresponds to the optimal pairing, and let d^* be the optimal d value. We call a cell (i, j) *admissible* if $u_{ij} \leq d^*$. The problem of listing all the optimal assignments then becomes the problem of listing all possible assignments that use only admissible cells. Using the flow chart of Fig. 3 can accomplish this. The number of operations can be seen to be dependent on the order of the columns of U. The dependence is quite complicated. However, a good rule of thumb for reducing the number of operations is to rearrange the columns in ascending order according to the number of their admissible cells.

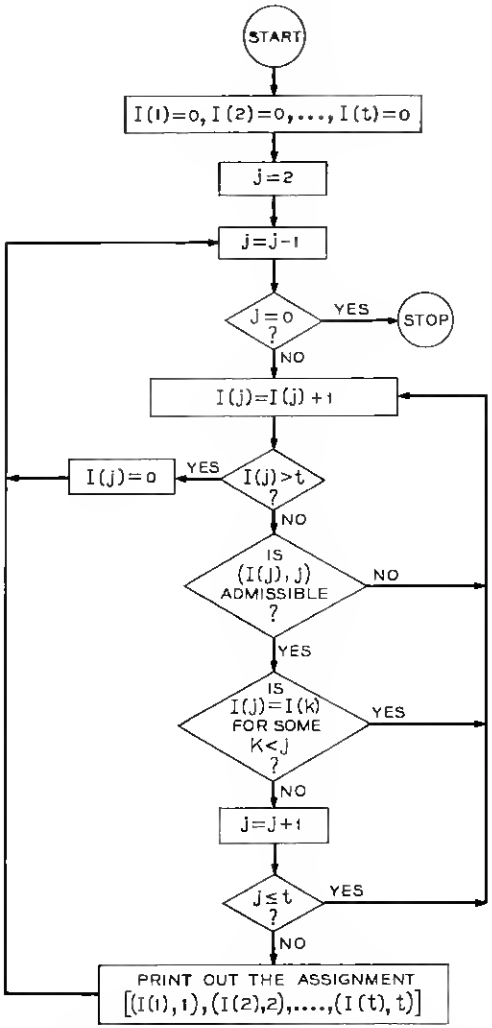

Fig. 3 — Flow chart for generating all optimal solutions.

REFERENCES

1. Lueder, E., "Cascading of RC-Active Two-Ports in Order to Minimize Inband Losses and to Avoid Distortion," Proceedings of the International Conference on Communications, Boulder, Colorado, June 9–11, 1969.
2. Gross, O., "The Bottleneck Assignment Problem," The Rand Corporation, Paper P-160, March 6, 1959.